In this issue

3  REXX extensions
9  Multiplatform command scheduler – part 4
19  Horizontal prefix line for manipulating columns
28  CMS back-up/restore – part 5
42  Dynamic menus system for CMS
52  VM news

© Xephon plc 1997
VM Update

Published by
Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 38342
From USA: 01144 1635 38342
E-mail: xephon@compuserve.com

North American office
Xephon/QNA
1301 West Highway 407, Suite 201-405
Lewisville, TX 75067
USA
Telephone: 940 455 7050

Australasian office
Xephon/RSM
GPO Box 6258
Halifax Street
Adelaide, SA 5000
Australia
Telephone: 088 223 1391

Editor
Trevor Eddolls

Disclaimer
Readers are cautioned that, although the information in this journal is presented in good faith, neither Xephon nor the organizations or individuals that supplied information in this journal give any warranty or make any representations as to the accuracy of the material it contains. Neither Xephon nor the contributing organizations or individuals accept any liability of any kind howsoever arising out of the use of such material. Readers should satisfy themselves as to the correctness and relevance to their circumstances of all advice, information, code, JCL, EXECs, and other contents of this journal before making any use of it.

Subscriptions and back-issues
A year’s subscription to VM Update, comprising twelve monthly issues, costs £170.00 in the UK; $255.00 in the USA and Canada; £176.00 in Europe; £182.00 in Australasia and Japan; and £180.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1990 issue, are available separately to subscribers for £14.50 ($21.50) each including postage.

VM Update on-line
Code from VM Update can be downloaded from our Web site at http://www.xephon.com; you will need the user-id shown on your address label.

Editorial panel
Articles published in VM Update are reviewed by our panel of experts. Members of the panel include John Illingworth (UK), Reinhard Meyer (Germany), Philippe Taymans (Belgium), Romney White (USA), Martin Wicks (UK), and Jim Vincent (USA).

Contributions
Articles published in VM Update are paid for at the rate of £170 ($250) per 1000 words for original material. To find out more about contributing an article, without any obligation, please contact us at any of the addresses above and we will send you a copy of our Notes for Contributors.

© Xephon plc 1997. All rights reserved. None of the text in this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior permission of the copyright owner. Subscribers are free to copy any code reproduced in this publication for use in their own installations, but may not sell such code or incorporate it in any commercial product. No part of this publication may be used for any form of advertising, sales promotion, or publicity without the written permission of the publisher. Copying permits are available from Xephon in the form of pressure-sensitive labels, for application to individual copies. A pack of 240 labels costs $36 (£24), giving a cost per copy of 15 cents (10 pence). To order, contact Xephon at any of the addresses above.

Printed in England.
**REXX extensions**

When I create an SQL/DS table it is possible to define a column with a numeric data type of DECIMAL. This column will then contain packed numbers. To optimize the processing of extracted packed data, the following REXX extensions, defined in the external function package RXUSERFN, are created:

- **P2D** – converter from packed to decimal
- **D2P** – converter from decimal to packed.

These functions appear to the user like ordinary REXX functions.

RXUSERFN is written in Assembler. The extensions use CMS with VM/SP Release 5.

The size of RXUSERFN is 424 bytes. An additional nucleus storage space of 304 bytes is required to load the package as a nucleus extension.

**FUNCTIONS**

In order to provide high performance when compacting and archiving program code, no additional checks are made on function parameters. Invalid parameters can therefore cause errors and may cause a specification or protection exception.

The package is loaded once. If some function from the package is nucxdropped, it will not be automatically nucxloaded, and it will generate an error at subsequent CALLs. To avoid this situation for REXX, the following command should be issued

```
NUCXDROP RXUSERFN
```

The program Packed to Decimal (P2D) returns the decimal value of the string representation of the packed number. The maximum string length is 8 bytes and is considered to be an image of the positive packed number.

The length of the resulting string is:
2 * n - 1 bytes

where n is the length of the input string and is between 1 and 8 inclusive.

Here are some examples:

P2D('1C'X) -> 1
P2D('1D'X) -> 1
P2D('12C'X) -> Ø12
P2D('123C'X) -> 123
P2D('123456789012345C'X) -> 123456789012345

The program Decimal to Packed (D2P), returns a string, which is the packed representation of the decimal number. The ‘whole-number’ must be an unsigned number with up to 15 digits.

The length of the resulting string is:

(2 + n) % 2 bytes

where: n is the number of decimal digits of input constant and is between 1 and 15 inclusive; and % will divide and return the integer part of the result.

Here are some examples:

D2P(1) -> '1C'X
D2P(12) -> 'Ø12C'X
D2P(123) -> '123C'X
D2P(1234) -> 'Ø1234C'X
D2P(123456789012345) -> '123456789012345C'X

INSTALL EXEC

********************************************************************
/***                                                  ***         ***/
/*** INSTALL         generate RXUSERFN MODULE         ***  DG'97  ***/
/***                                                  ***         ***/
********************************************************************
/***   SIZE ØØØ43  VER 1.Ø MOD ØØ  TIME 16:57:35  DATE 11/Ø7/97   ***/
********************************************************************
CLRSCRN
MESSAGE = 'user request'
SAY ' --- Start RXUSERFN MODULE generation - reply Y or N'
PULL REPLY
IF REPLY ¬= 'Y' THEN
SIGNAL ERROR
SET CMSTYPE HT
STATE RXUSERFN MODULE A
SAVE_RC = RC
SET CMSTYPE RT
IF SAVE_RC = Ø THEN
  DO
    SAY ' --- RXUSERFN MODULE found on disk A'
    SAY ' --- Replace RXUSERFN MODULE A - reply Y or N'
    PULL REPLY
    IF REPLY ≠ 'Y' THEN
      SIGNAL ERROR
  END
END
SET CMSTYPE HT
SIGNAL ON ERROR
MESSAGE = 'error when assemble' RXUSERFN
ASSEMBLE RXUSERFN
ERASE RXUSERFN LISTING A
MESSAGE = 'error when load' RXUSERFN
LOAD RXUSERFN '(' NOMAP NOLIBE
MESSAGE = 'error when genmod' RXUSERFN
GENMOD
ERASE RXUSERFN TEXT A
SIGNAL OFF ERROR
SET CMSTYPE RT
SAY ' --- RXUSERFN MODULE generated successfully'
EXIT
ERROR:
  SET CMSTYPE RT
  SAY ' --- RXUSERFN MODULE not generated due to' MESSAGE

RXUSERFN ASSEMBLE

******************************************************************************
****                                                  ***         ****
**** RXUSERFN              REXX extension             ***  DG'97  ****
****                                                  ***         ****
******************************************************************************
****   SIZE Ø9154  VER 1.0 MOD ØØ  TIME 17:05:13  DATE 11/Ø7/97   ****
******************************************************************************
*                                                                    *
RXUSERFN CSECT
  USING *,12
  LR  11,14
  CLC 8(8,1),=CL8'LOAD'
  BE DONUCEXT
  RET
  EQU *
  LA  15,1
  BR  11
DONUCEXT EQU *
LA 3,EXTS
LA 4,12
LA 5,EXTE
L Ø,TOALLOC
LR 10,0
DMSFREE DWORDS=(Ø),TYPE=NUCLEUS,ERR=RET
LR 0,1
LR 1,10
SLL 1,3
ST 1,NUCXLEN
LA 14,USERBGN
LR 15,1
LR 10,0
SPKA Ø
MVCL Ø,14
LR 0,10
B SETXBLK

NEXTEXT EQU *
XC NUCXLEN(4),NUCXLEN
L Ø,0(3)
AR Ø,10

SETXBLK EQU *
ST Ø,NUCXADDR
ST Ø,NUCXORG
MVC NUCXNAME(8),4(3)
LA 1,NUCXDCL
SVC 2Ø2
DC AL4(1)
LTR 15,15
BNZ RET
BXLE 3,4,NEXTEXT
BR 11

EXTS DS 0F
USEROFF DC A(Ø)
USERNM DC CL8'RXUSERFN'
P2DOFF DC A(P2DBGN-USERBGN)
P2DNM DC CL8'P2D'
D2POFF DC A(D2PBGN-USERBGN)
D2PNM DC CL8'D2P'
EXTE EQU *-12
TOALLOC DC A((REALEND-USERBGN+7)/8)
NUCXDCL DS 0F
DC CL8'NUCEXT'
NUCXNAME DS CL8
NUCXSM DC X'ØØ'
NUCXCMWP DC X'Ø4'
NUCXFLG DC X'80'
USERFLG DC X'00'
NUCXADDR DS A
USERWORD DC A(Ø)
NUCXORG DS A
NUCXLEN DS A
LTO RG
USERBGN DS ØD
LA 15,1
BR 14
DROP 12
P2DBGN DS ØD
USING *,12
LR 11,14
LR 10,0
USING EPLIST,1Ø
L 1,ARGLIST
L 2,4(1)
LR 3,2
SLL 3,1
BCTR 3,Ø
BCTR 2,0
STC 2,UNPK+1
OI UNPK+1,X'FØ'
L 1,Ø(1)
UNPK UNPK VALUE(Ø),Ø(Ø,1)
OI VALUE+15,X'FØ'
B EPILOG
P2DEND EQU *
DROP 12
D2PBGN DS ØD
USING *,12
LR 11,14
LR 10,0
USING EPLIST,1Ø
L 1,ARGLIST
L 2,4(1)
LA 3,2(2)
SRL 3,1
BCTR 2,0
STC 2,PACK+1
OI PACK+1,X'FØ'
L 1,Ø(1)
PACK PACK VALUE(Ø),Ø(Ø,1)
NI VALUE+15,X'FØ'
OI VALUE+15,X'ØC'
B EPILOG
D2PEND EQU *
DROP 12
EPILOG EQU *
BALR 12,0
USING *,12
LR 2,3
LA    3.23(3)
SRL   3.3
LR    0.3
DMSFREE DWORDS=(Ø),TYPE=NUCLEUS,ERR=ERR
L    14,SYSFUNRT
ST    1,Ø(14)
USING EVALBLK,1
XC EVBPAD2(4),EVBPAD2
ST    2,EVLEN
ST    3,EVSIZE
BCTR  2,Ø
STC   2,MVC+1
ICM   3,7,MVC+3
LA    4,15
SR    4,2
LR    2,3
AR    3,4
STCM  3,7,MVC+3
MVC MVC EVDATA,VALUE
STCM  2,7,MVC+3
ERR   EQU *
BR    11
VALUE DS CL16
REALEN EQU *
EPLIST DSECT
COMVERB DS A
BEGINARGS DS A
ENDARGS DS A
FBLOK DS A
ARGLIST DS A
SYSFUNRT DS A
EVALBLK DSECT
EVBPAD1 DS F
EVSIZE DS F
EVLEN DS F
EVBPAD2 DS F
EVDATA DS C
END RXUSERFN

GETTING READY

The INSTALL EXEC should be used to generate the nucleus extension code. No additional actions are needed. The functions must be called from EXECs in the same way as normal REXX built-in functions.

Dobrin Goranov
Information Services Co (Bulgaria) © Dobrin Goranov 1997
This month we conclude the code that allows multiplatform command scheduling.

\[
\text{THISYEAR} = \text{TRANSLATE(FORMAT(LEFT(DATE('O'),2),2),'Ø',' ')} \\
\text{LEAPYEAR} = ((\text{THISYEAR}//4) = 0) \quad */ \text{IS THIS A LEAP YEAR} */ \\
\text{IF LEAPYEAR THEN YEARDAYS} = 366; \text{ELSE YEARDAYS} = 365 \\
\text{IF LEFT(IDATEX,1) = '*' THEN IDATEX = DATE('U') \quad */ FMT = MM/DD/YY */} \\
\text{IDATEL = LENGTH(IDATEX)} \\
\text{IF IDATEL < 3 THEN SIGNAL ERR020} \\
\text{IF IDATEL > 10 THEN SIGNAL ERR030} \\
\text{IF IDATEL = 3 \& IDATEX > YEARDAYS THEN SIGNAL ERR050} \\
\text{IF IDATEL = 3 THEN IDATEX = THISYEAR||IDATEX} \\
\text{IDATEL = LENGTH(IDATEX)} \\
\text{IF IDATEL = 4 THEN IDATEX = IDATEX||THISYEAR \quad */ MMDD */} \\
\text{IDATEL = LENGTH(IDATEX)} \\
\text{IF IDATEL = 6 \& SUBSTR(IDATEX,3,1) = '/' \&.} \\
\text{POS('/',SUBSTR(IDATEX,4)) = 0 \quad */ YY/DDD */} \\
\text{THEN IDATEX = LEFT(IDATEX,2)RIGHT(IDATEX,3) \quad */ YYDDD */} \\
\text{IDATEL = LENGTH(IDATEX)} \\
\text{IF IDATEL = 5} \\
\text{THEN IF SUBSTR(IDATEX,3,1) = '/' \quad */ MM/DD */} \\
\text{THEN IDATEX = SUBSTR(IDATEX,1,2)SUBSTR(IDATEX,4,2)THISYEAR \quad */ =}> \\
\text{MMDDYY */} \\
\text{IDATEL = LENGTH(IDATEX)} \\
\text{IF (IDATEL > 2 \& IDATEL < 8) \&,} \\
\text{DATATYPE(IDATEX) \rightleftharpoons 'NUM'} \\
\text{THEN SIGNAL ERR040} \\
\text{IF IDATEL = 5} \\
\text{THEN DO} \\
\text{JDATE = IDATEX} \\
\text{YYEAR = LEFT(JDATE,2)} \\
\text{JLEAP = ((YYEAR)//4) = 0} \\
\text{IF JLEAP THEN MDAYS.2 = 29} \\
\text{JDAYS = RIGHT(JDATE,3)} \\
\text{MDAYS = MATHYEAR(YYEAR) + JDAYS} \\
\text{PARSE VALUE JULTOMON(JDAYS) WITH MMONTH DDAY.} \\
\text{SDATE = YYEAR||MMONTH||DDAY} \\
\text{GDATE = MMMONTH||DDAY||YYEAR} \\
\text{FDATE = MMMONTH'/'DDAY'/'YYEAR} \\
\text{END} \\
\text{IF IDATEL < 8 THEN IF DATATYPE(IDATEX) \rightleftharpoons 'NUM' THEN SIGNAL ERR080} \\
\text{IF IDATEL = 6 \quad */ MMDDYY */} \\
\text{THEN DO} \\
\text{XX = LEFT(IDATEX,2)} \\
\text{*/ IF XX = 0 \mid XX > 12 */}
IF XX > 12
   THEN GDATE = RIGHT(IDATEX,4)||XX /* MMDDYY */
   ELSE GDATE = IDATEX
SDATE = RIGHT(GDATE,2)||LEFT(GDATE,4) /* YYMMDD */
FDATE = LEFT(GDATE,2)="/"SUBSTR(GDATE,3,2)="/"RIGHT(GDATE,2)
YYEAR = RIGHT(GDATE,2)
GLEAP = ((YYEAR)//4) = Ø
IF GLEAP THEN MDAYS.2 = 29
JDAYS = RIGHT('ØØ'|MONTOJUL(LEFT(GDATE,2))+SUBSTR(GDATE,3,2)),3)
MDAYS = MATHYEAR(YYEAR) + JDAYS
JDATE = YYEAR||JDAYS
END

MATHCALC:
IF IDATEL = 7 /* 1MMMMMM */
   THEN DO
      MDAYS = IDATEX - 1000000
      JDATE = MATHTOJULN(MDAYS)
      YYEAR = LEFT(JDATE,2)
      JDAYS = RIGHT(JDATE,3)
      MLEAP = ((YYEAR)//4) = Ø
      IF MLEAP THEN MDAYS.2 = 29
      PARSE VALUE JULTOMON(JDAYS) WITH MMONTH DDAY.
      SDATE = YYEAR||MMONTH||DDAY
      GDATE = MMONTH||DDAY||YYEAR
      FDATE = MMONTH"/"DDAY"/"YYEAR
      END
   IF IDATEL = 8 | IDATEL = 10 /* MM/DD/YY OR MM/DD/YYYY */
   THEN DO
      IF DATATYPE(SUBSTR(IDATEX,3,1)) = 'NUMERIC'
         THEN DO
            IF DATATYPE(RIGHT(IDATEX,4)) = 'NUM'
               THEN DO
                  /* CHOP OF YEAR PFX & MAKE IT 8 BYTES... */
                  IDATEX = LEFT(IDATEX,6)||RIGHT(IDATEX,2)
                  IDATEL = 8
               END
            IF DATATYPE(LEFT(IDATEX,4)) = 'NUM'
               THEN DO
                  /* CHOP OF YEAR PFX & MAKE IT 8 BYTES... */
                  IDATEX = SUBSTR(IDATEX,3)
                  IDATEL = 8
               END
            XX = LEFT(IDATEX,2)
            /* IF XX = Ø | XX > 12 */
            IF XX > 12
               THEN FDATE = RIGHT(IDATEX,5)||XX /* MM/DD/YY */
               ELSE FDATE = IDATEX
            GDATE = LEFT(FDATE,2)||SUBSTR(FDATE,4,2)||RIGHT(FDATE,2)
            YYEAR = RIGHT(GDATE,2)
            FLEAP = ((YYEAR)//4) = Ø
IF FLEAP THEN MDAYS.2 = 29
SDATE = RIGHT(FDATE,2)||LEFT(FDATE,2)||SUBSTR(FDATE,4,2)
IF DATATYPE(GDATE) = 'NUM' THEN SIGNAL ERR060
JDAYS=RIGHT('00'||(MONTJOUL(LEFT(GDATE,2))+SUBSTR(GDATE,3,2)),3)
MDAYS = MATHYEAR(YYEAR) + JDAYS
JDATE = YYEAR||JDAYS
END
ELSE SIGNAL ERR070
END

/*TRACE I*/
IF IDATEL > 8 THEN SIGNAL ERR030
MWEEK = TRANSLATE((MDAYS + 1) // 7,'7','0')
WDATE = DTEXT.MWEEK
MMONTH = LEFT(GDATE,2) + 0
MLEAP = ((YYEAR)//4) = 0
IF MMONTH = 0
THEN MDATE = MTEXT.MMONTH
ELSE MDATE = MTEXT.1
XDAYS = MDAYS + 1000000
/* Bugg... DOING ADJUSTMENTS OVER MULTIPLE YEARS THAT INCLUDE LEAPS GET OUT OF SYNC BY THE NUMBER OF LEAP YEARS TRAVERSED. */
IF IDATEADJ = '' & DATATYPE(IMATHADJ) = 'NUM' THEN DO
IDATEADJ = ''
ADJLEAP = ((IYEARADJ)//4) = 0
ADJDAYS = ((IMATHADJ - 1000000) - (MATHBASE - 1))
IF MLEAP & IADJTYPE = '+' THEN ADJDAYS = ADJDAYS + 1
IF IADJTYPE = '-' THEN DO
IF ADJLEAP & ADJDAYS < 366 THEN ADJDAYS = ADJDAYS + 1
IF ¬ADJLEAP & ADJDAYS < 365 THEN ADJDAYS = ADJDAYS + 1
END
/*IF ADJLEAP & IADJTYPE = '-' THEN ADJDAYS = ADJDAYS + 1 */
IF IADJTYPE = '+'
THEN XDAYS = XDAYS + ADJDAYS
ELSE XDAYS = XDAYS - ADJDAYS
IDATEL = LENGTH(XDAYS)
IDATEX = XDAYS
MDAYS = MATHCALC
MDAYS.2 = 28
/* TRACE O */
SIGNAL MATHCALC
/* GO BACK AND CALC DATE WITH ADJ VALUE INCLUDED. */
END

***************************************************************************/
EXIT:
IF ¬MSGONLY
THEN PUSH '*' FDATE XDAYS GDATE SDATE,
JDATE WDATE MDATE '19'YYEAR MWEEK
IF ¬QUIET
THEN SAY 'REXXDATE -' FDATE XDAYS GDATE SDATE,
JDATE WDATE MDATE '19'YYEAR MWEEK
EXIT ØØØ
/************************************************************************************/
/************************************************************************************/
MATHYEAR: PROCEDURE EXPOSE MATHBASE
PARSE ARG YY.
MATHYEAR = (((YY * 365) + (YY % 4) + (MATHBASE - 1))
IF (YY//4) = 0 THEN MATHYEAR = MATHYEAR - 1
/* COUNT CENTURIES NOT DIVISIBLE BY 400 AND SUBTRACT. */
RETURN MATHYEAR
/************************************************************************************/
YEARMATH: PROCEDURE EXPOSE MATHBASE
PARSE ARG NN.
YEARMATH = (((NN - (NN % 4*365) - (MATHBASE - 1)) / 365) % 1
IF MATHYEAR(YEARMATH) > NN THEN YEARMATH = YEARMATH - 1
/* COUNT CENTURIES NOT DIVISIBLE BY 400 AND SUBTRACT. */
RETURN YEARMATH
/************************************************************************************/
MATHTOJULN: PROCEDURE EXPOSE MATHBASE
PARSE ARG MATHDATE.
YLEAPS = (((MATHDATE - (MATHBASE-1)) / (365 * 4)) + 0.997) % 1
ZLEAPS = (((MATHDATE - (MATHBASE-1)) % (365 * 4))
YYEAR = RIGHT('0'||((((MATHDATE-(MATHBASE-1)-YLEAPS)/365) + 0.003) %
1),2)
/*
JJJULN = RIGHT('00'||((MATHDATE - (MATHBASE-1) - YLEAPS) // 365),3)
*/
IF MATHYEAR(YYEAR) >= MATHDATE THEN YYEAR = YYEAR - 1
JJJULN = RIGHT('00'||(MATHDATE - MATHYEAR(YYEAR)),3)
JLEAP = (YYEAR // 4) = 0
IF JJJULN = 0
THEN DO
IF JLEAP /* IS THIS A LEAP YEAR */
THEN JJJULN = 366
ELSE JJJULN = 365
END
ELSE DO
/*IF JLEAP & JJJULN > 60
THEN JJJULN = RIGHT('00'||(JJJULN + 1),3) */
END
/* COUNT CENTURIES NOT DIVISIBLE BY 400 AND SUBTRACT. */
RETURN YYEAR||JJJULN
/************************************************************************************/
MONTOJUL: PROCEDURE EXPOSE MDAYS.
PARSE ARG MONTH.
DAYS = 0; IF NOT(MONTH < 1 | DATATYPE(MONTH) == 'NUM' | MONTH > 12)
THEN DO NDX = 1 WHILE NDX < MONTH
  DAYS = DAYS + MDAYS.NDX
END
MONTOJULX:
JULNDAYS = RIGHT('000'DAYS,3)
RETURN JULNDAYS

******************************************************************************
JULTOMON: PROCEDURE EXPOSE MDAYS.
PARSE ARG JULNDAYS.
IF JULNDAYS <= 0 | DATATYPE(JULNDAYS) = 'NUM'
  THEN DO /* DEFENSIVE CODING LOGIC A */
    MONTH = 1; DAYS = 0
    SIGNAL JULTOMONX
  END
IF (MDAYS.2 = 28 & JULNDAYS > 365) |,
(MDAYS.2 = 29 & JULNDAYS > 366)
  THEN DO /* DEFENSIVE CODING LOGIC B */
    MONTH = 12
    DAYS = (JULNDAYS - (337 + MDAYS.2))
    SIGNAL JULTOMONX
  END
DO NDX = 1 WHILE JULNDAYS /= 0
  IF JULNDAYS > MDAYS.NDX
    THEN DO
      JULNDAYS = JULNDAYS - MDAYS.NDX
    END
  ELSE DO
    MONTH = NDX
    DAYS = JULNDAYS
    JULNDAYS = 0
  END
END
JULTOMONX:
MONSDAYS = RIGHT('00'MONTH,2) RIGHT('00'DAYS,2)
RETURN MONSDAYS

FIND: PROCEDURE
PARSE ARG STR,FND
POS = WORDPOS(FND,STR)
RETURN POS

******************************************************************************
DOC:
/*BEGTYPE
REXXNAME: REXXDATE
FUNCTION: GENERATE A STRING OF CURRENT OR ENTERED DATE IN USEFUL
FORMATS.
FORMAT: ENTER AS SHOWN BELOW:

  REXXDATE &BASDATE < &ADJDATE > <*QUIET> <*MSGONLY>
  &BASDATE MEANS ENTER THE BASE DATE IN ONE OF THE FOLLOWING
  FORMATS.

*|FULLDAT|MATHDAT|GREGDAT|SORTDAT|JULNDAT
&ADJDATE MEANS ENTER ADJUSTMENT DATE IN ONE OF THE FOLLOWING FORMATS. THE ADJUSTMENT DATE IS AN OPTIONAL FIELD THAT ALLOWS THE USER TO ADD OR SUBTRACT THE NUMBER OF DAYS, MONTHS, OR YEARS FROM THE ENTERED BASE DATE. FOR EXAMPLE, BASE DATE OF Ø1/Ø1/94 FOLLOWED BY AN ADJUSTMENT DATE OF +Ø1Ø31 WILL GENERATE ALL THE DATE VALUES FOR Ø2/Ø1/95.
<+|->FULLDAT|MATHDAT|GREGDAT|SORTDAT|JULNDAT
| MEANS SELECT ONE OF THE FORMATS SHOWN.
< > MEANS FIELD WITHIN IS OPTIONAL.
* GENERATE SET OF DATE FMTS FOR TODAY'S DATE.
FULLDAT ENTER FULL DATE, 8 BYTES LONG, IN FMT MM/DD/YY.
MATHDAT ENTER MATH DATE, 7 BYTES LONG FORMATED USING FORMULA:
FORMULA=(JUL-DAY)+(YR X 365)+(YR/4(IF EVEN SUB 1))+1000000.
GREGDAT ENTER GREGORIAN DATE, 6 BYTES LONG, IN FMT MMDDYY.
IF YEAR GREATER THAN 12 YOU MAY USE SORTDATE FORMAT YYMMDD.
SORTDAT ENTER SORT DATE, 6 BYTES LONG, IN FMT YYMMDD.
JULNDAT ENTER JULIAN DATE, 5 BYTES LONG, IN FMT YYDDD.
QUIET OPTIONAL, SUPPRESSES TERMINAL DISPLAY OF DATE INFO.

OUTPUT: THIS EXEC WILL STACK THE FOLLOWING LINE IN YOUR TERMINAL BUFFER.

* &FULLDAT &MATHDAT &GREGDAT &SORTDAT &JULNDAT &DOWFWRK &MONOFYR &YR &NUMDAY

EXAMPLES: IF THIS WERE RUN ON Ø9/1Ø/81 THE FOLLOWING WOULD HAPPEN.
LINE WITH STAR(*) IS WHAT GETS PUT INTO THE STACK.

REXXDATE * *QUIET
* Ø9/1Ø/81 1723432 Ø91Ø81 81Ø91Ø 81253 THURSDAY OCTOBER 1981 4

IF YOU WANTED TO SEE WHAT THE DATE WOULD BE 25 DAYS INTO THE FUTURE ENTER THE FOLLOWING. USE *TEST OPTION TO NOT STACK RESULT.
ASSUME SAMPLE WAS RUN ON 11/29/95.

REXXDATE * +25 *TEST
REXXDATE - ØØ/25/ØØ 1693618 ØØ25ØØ ØØØØ25 ØØØ25 WEDNESDAY JANUARY 19ØØ 3
REXXDATE - 12/23/95 1728649 122395 122395 95357 SATURDAY DECEMBER 1995 6

EXIT ØØØ
ERR010:
SAY 'REXXDATE - INVALID NUMBER OF INPUT PARAMETERS. SEE DOC.'
EXIT 010
ERR020:
SAY 'REXXDATE - DATE FIELD MUST BE AT LEAST 3 BYTES LONG. SEE DOC.'
EXIT 020
ERR030:
SAY 'REXXDATE - DATE FIELD CAN NOT BE OVER 8 BYTES LONG. SEE DOC.'
EXIT 030
ERR040:
SAY 'REXXDATE - DATE FIELD IS INVALID, FOUND ('IDATEX').'
EXIT 040
ERR050:
SAY 'REXXDATE - JULIAN DAY DATE FIELD IS INVALID, FOUND ('IDATEX').'
EXIT 050
ERR060:
SAY 'REXXDATE - GREGORIAN DATE FIELD IS INVALID FOR ('IDATEX').'
EXIT 060
ERR070:
SAY 'REXXDATE - FULL 8 BYTE DATE FIELD IS INVALID FOR ('IDATEX').'
EXIT 070
ERR080:
SAY 'REXXDATE - ' IDATEL ' BYTE LONG DATE FIELDS MUST BE FULLY NUMERIC.'
SAY 'FOUND ('IDATEX').'
EXIT 080
ERR090:
SAY 'REXXDATE - THE CONVERSION OF THE ADJUSTMENT DATE INTO A MATH DATE'.
'FAILED.'
SAY 'FAILED COMMAND WAS: "DATEREXX' IDATEADJ XQUIET'"'
EXIT 090

```h
TIMECALC EXEC

/ * * /
PARSE UPPER ARG ARGSTRING; DEBUG = ''
IF (FIND(ARGSTRING,'?')) = 1 THEN SIGNAL DOC
X = (FIND(ARGSTRING,'*DEBUG'))
IF X = Ø THEN DO
ARGSTRING = (DELWORD(ARGSTRING,X,1)); TRACE (I); DEBUG = '*DEBUG'
END
QUIETOPT = ''; X = (FIND(ARGSTRING,'*QUIET'))
IF X = Ø THEN X = FIND(ARGSTRING,'QUIET')
IF X = Ø THEN DO
ARGSTRING = (DELWORD(ARGSTRING,X,1)); QUIETOPT = 'QUIET'
END
TESTOPT = Ø; X = (FIND(ARGSTRING,'*TEST'))
IF X = Ø THEN DO
ARGSTRING = (DELWORD(ARGSTRING,X,1)); TESTOPT = 1
```

BEGIN:
/* TIMECALC CTLSECS PLUSHRS PLUSMINS PLUSSECS */
CTLSECS = SUBWORD(ARGSTRING,1,1)
PLUSHRS = SUBWORD(ARGSTRING,2,1)
IF POS('::',CTLSECS) ≠ ø & PLUSHRS = '' THEN DO
    plushrs = cTLSecs
cTLSecs = ø
END
IF POS('::',PLUSHRS) ≠ ø THEN DO
    parse value plushrs with plushrs '::' plusmins '::' plussecs
    parse var plussecs plussecs '.'
signal setdflts
END
PLUSMINS = SUBWORD(ARGSTRING,3,1)
PLUSSECS = SUBWORD(ARGSTRING,4,1)
SETDFLTS:
    if CTLSECS = '' THEN CTLSECS = 00000
    if CTLSECS = '*' THEN CTLSECS = TIME(S)
    if plushrs = '' | plushrs = '*' THEN plushrs = 00000
    if plusmins = '' | plusmins = '*' THEN plusmins = 00000
    if plussecs = '' | plussecs = '*' THEN plussecs = 00000
secs = ø; mins = ø; hrs = ø; days = ø; plusedays = ø
    if DATATYPE(CTLSECS) ≠ 'NUM' THEN SIGNAL ERR010
    if DATATYPE(PLUSHRS) ≠ 'NUM' THEN SIGNAL ERR020
    if DATATYPE(PLUSMINS) ≠ 'NUM' THEN SIGNAL ERR030
    if DATATYPE(PLUSSECS) ≠ 'NUM' THEN SIGNAL ERR040
    if plussecs > 59 THEN mins = SUBSTR(FORMAT((PLUSSECS/60),4,4),1,4)
    if plussecs > 59 THEN PLUSSECS = (PLUSSECS - (MINS*60))
    PLUSMINS = PLUSMINS + MINS
    if plusmins > 59 THEN HRS = SUBSTR(FORMAT((PLUSMINS/60),3,4),1,3)
    if plusmins > 59 THEN PLUSMINS = (PLUSMINS - (HRS*60))
    plushrs = plushrs + hrs
    if plushrs > 23 THEN DAYS = SUBSTR(FORMAT((PLUSHRS/24),3,4),1,3)
    if plushrs > 23 THEN PLUSHRS = (PLUSHRS - (DAYS*24))
    PlUSDAYS = PlUSDAYS + DAYS
Newdays = ø; newhrs = ø; Newmins = ø; newsecs = ø
/* say CTLSECS DAYS HRS MINS SECS */
Newtime = CTLSECS + PLUSSECS + (PLUSMINS*60) +,
    (PLUSHRS*60*60) + (PlUSDAYS*24*60*60)
Newtimex = Newtime
If Newtimex ¬= ø
    THEN NEWDAYS = SUBSTR(FORMAT((NEWTIMEX/(86400)),3,4),1,3)
NEWTIMEX = (NEWTIMEX - (NEWDAYS*86400))
IF NEWTIMEX = 0 THEN NEWHRS = SUBSTR(FORMAT((NEWTIMEX/(3600)),3,4),1,3)
NEWTIMEX = (NEWTIMEX - (NEWHRS*3600))
IF NEWTIMEX = 0 THEN NEWMINS = SUBSTR(FORMAT((NEWTIMEX/60),3,4),1,3)
NEWTIMEX = (NEWTIMEX - (NEWMINS*60))
IF NEWTIMEX = 0 THEN NEWSECS = FORMAT((NEWTIMEX),3,0)
FMTDTIME = RIGHT((NEWHRS),2)||':'||RIGHT((NEWMINS),2),
||':'||RIGHT((NEWSECS),2)
FMTDTIME = TRANSLATE(FMTDTIME,'0',' ') 
IF QUIETOPT = 'QUIET'
THEN SAY 'NEWTIME = ' NEWTIME 'IN SECONDS. NEW DD HH MM SS = ',
       NEWDAYS NEWHRS NEWMINS NEWSECS FMTDTIME||'.'
IF TESTOPT THEN SIGNAL EXIT
PUSH '**' NEWTIME NEWDAYS NEWHRS NEWMINS NEWSECS FMTDTIME
EXIT:
EXIT 000
DOC:
'VMFCLEAR'
SAY 'REXXNAME: TIMECALC'
SAY 'FUNCTION: TO PROVIDE A SIMPLE SECONDS IN DAY INCREMENT CALCULATOR.'
SAY 'FORMAT: ENTER COMMAND AS SHOWN BELOW.'
SAY 'TIMECALC < *|&CTLSECS &ADDHRS &ADDMINS &ADDSECS <QUIET>>'
SAY '< > DO NOT CODE THESE. THEY MEAN FIELDS WITHIN ARE
OPTIONAL.'
SAY '*|&CTLSECS THIS FIELD IS THE CONTROL NUMBER OF SECONDS. IT IS USED'
SAY 'TO CALCULATE A STARTING DAY/HH/MM/SS VALUE. IF A "*" IS'
SAY 'ENTERED THE CURRENT TIME BECOMES THE CONTROL.'
SAY '&ADDHRS THIS FIELD IS A NUMBER OF RAW HOURS BETWEEN 0 AND 1000'
SAY 'IT IS CONVERTED TO SECONDS AND ADDED TO &CTLSECS TO '
SAY 'PROVIDE A NEW &CTLSECS BASE TO CALCULATE DAY/HH/MM/SS.'
SAY 'THIS FIELD IS OPTIONAL, BUT IF CODED THE PRIOR FIELD'
SAY 'MUST BE CODED TOO.'
SAY '&ADDMINS THIS FIELD IS NUMBER OF RAW MINUTES BETWEEN 0 AND 1000'
SAY 'IT IS CONVERTED TO SECONDS AND ADDED TO &CTLSECS TO '
SAY 'PROVIDE A NEW &CTLSECS BASE TO CALCULATE DAY/HH/MM/SS.'
SAY 'THIS FIELD IS OPTIONAL, BUT IF CODED ALL PRIOR FIELDS'
SAY 'MUST BE CODED TOO.'
SAY '&ADDSECS THIS FIELD IS NUMBER OF RAW SECONDS BETWEEN 0 AND 1000'
SAY 'IT IS CONVERTED TO SECONDS AND ADDED TO &CTLSECS TO '
SAY 'PROVIDE A NEW &CTLSECS BASE TO CALCULATE DAY/HH/MM/SS.'
SAY 'THIS FIELD IS OPTIONAL, BUT IF CODED ALL PRIOR FIELDS'
SAY 'MUST BE CODED TOO.'
SAY 'QUIET THIS EXEC TYPES A MESSAGE TO THE USER DISPLAYING THE '
SAY 'RESULTS OF ITS CALCULATIONS, THAT WERE PUT INTO THE STACK.'
SAY '" IF THE USER DOESN'T WANT TO SEE THIS MESSAGE THEY CAN"
SAY 'ENTER QUIET AS THE LAST FIELD, AND HAVE IT SUPPRESSED.'
SAY 'THIS FIELD IS OPTIONAL, BUT IF CODED ALL PRIOR FIELDS'
SAY 'MUST ALSO BE CODED. '
SAY 'STACKFMT: BELOW IS THE FORMAT OF THE OUTPUT OF TIMECALC. IT IS'
SAY 'PUT INTO THE CONSOLE STACK.'
SAY '* &NEWBASE &NEWDAYS &NEWHH &NEWMM &NEWSS &FMTDTIME'
SAY '&NEWBASE  THIS IS THE FINAL TIME VALUE IN SECONDS AFTER ALL ' 
SAY '&NEWBASE  HOURS, MINUTES, AND SECONDS HAVE BEEN ADDED TOGETHER. '
SAY '&NEWDAYS  THIS IS THE NUMBER OF DAYS REMAINING, INTO THE FUTURE' 
SAY '&NEWDAYS  WHEN THE HOURS, MINUTES, AND SECONDS ENTERED GO BEYOND'
SAY '&NEWDAYS  THE CURRENT DAY. IF NO DAY SPILLOVER VALUE IS ZERO.'
SAY '&NEWXX   THIS IS THE FINAL TIME VALUE IN HOURS, MINUTES, OR'
SAY '&NEWXX   SECONDS BASED ON NORMAL CLOCK TIMES OF 24, 60, AND 60.'
SAY '&FMTDTIME THIS IS THE FINAL TIME VALUE IN THE STANDARD TIME '
SAY '&FMTDTIME  FORMAT OF "HH:MM:SS".'
SAY 'EXAMPLES: THE EXAMPLE BELOW WILL CALCULATE THE TIME TEN DAYS FROM'
SAY 'WHEN IT IS EXECUTED, AND NOT DISPLAY ANYTHING TO USER.'
SAY 'TIMECALC * 24Ø Ø Ø QUIET'
SAY 'BELOW IS AN EXAMPLE OF WHAT MIGHT BE STACKED AND SHOWN'
SAY 'EXAMPLE OF TEN DAYS WAS USED. FIRST THE DISPLAY MESSAGE'
SAY 'IS DISPLAYED, THEN THE STACK VALUE.'
SAY '* NEWTIME = 9Ø54ØØ IN SECONDS.',
    'NEW DD HH MM SS = 1Ø 11 3Ø Ø 11:3Ø:ØØ.'
SAY '* 9Ø54ØØ 1Ø 11 3Ø Ø 11:3Ø:ØØ.'
SAY EXIT Ø0Ø
ERRØ1Ø:
SAY 'TIMECALC - THE CTLSECS IS NOT NUMERIC. FOUND (' CTLSECS ').'
EXIT Ø1Ø
ERRØ2Ø:
SAY 'TIMECALC - THE PLUSHRS IS NOT NUMERIC. FOUND (' PLUSHRS ').'
EXIT Ø2Ø
ERRØ3Ø:
SAY 'TIMECALC - THE PLUSMINS IS NOT NUMERIC. FOUND (' PLUSMINS ').'
EXIT Ø3Ø
ERRØ4Ø:
SAY 'TIMECALC - THE PLUSSECS IS NOT NUMERIC. FOUND (' PLUSSECS ').'
EXIT Ø4Ø

Marc Vincent Irvin
Move Immediate Software (USA)  © M V Irvin 1997
Horizontal prefix line for manipulating columns

The normal prefix area in XEDIT allows you to manipulate rows in a very straight-forward manner. So why shouldn’t we do the same with columns? To this end, I wrote HP XEDIT, which allows us to insert, delete, move, copy, and duplicate columns or specify some column-related settings. The commands are not quite the same as in row manipulation – some prefix commands were added to set zone, truncation, and even the ‘current’ column.

If you need this feature, just call HP from inside XEDIT without parameters. If there is already a SCALE, this line is superseded by another line which looks very similar to the scale line, where you can enter your column subcommands by overtyping.

You can reset the horizontal prefix area by calling HP XEDIT once more – it works like a toggle.

More than one subcommand can be entered at once. Especially when using the move and copy subcommands, the target column must be entered at the same time (with P for preceding and F for following).

Block subcommands are supported (ie dd..dd, cc..cc, mm..mm, and "..") in the same way as in the normal prefix area.

Only lines beginning with the current line are manipulated by HP subcommands. One important point – HP actually replaces ALL lines from the rest of the file by deleting and inserting the new ones! So do not try to use HP subcommands when you have excluded lines (eg with the ALL/../ command)!

HP XEDIT does not consider all the possible XEDIT settings. Some important points:

• I did not find any way to make HP XEDIT work properly with a split screen containing more than one file in a ring. You can try it, but commands are accepted only for the file where HP is active. (Remember, you can reset the HP area by calling HP a second time!)

• I did not even try to consider special settings like ‘SET VERIFY
1 10 H 20 30’, tabs, windows, split screens, and excluded lines, which interfere with other macros that use ‘READ ALL TAG’ and many other possible settings.

- Pending subcommands are not supported – the target column for move and copy subcommands (M and C) must be entered at the same time as the subcommand itself.
- If there is a scale line, it is replaced by a RESERVED line with input capabilities, otherwise this prefix area is set at line 3. (You can change this – see comments in HP XEDIT.)
- Up to now I have not implemented a syntax like ‘D6’ to delete 6 columns at once (maybe another time). Instead use block
commands using a syntax as in the following example to delete columns 13 to 18:

‘...+....1..dd+.dd.2....+’

- Switch off HP as soon as you don’t need it any more – it will help your performance.
- If you have a REXX compiler, use it!
- HP XEDIT is *not* foolproof, but it works fairly well.
- All subcommands supported are shown in the comments in HP

```plaintext
make column 6 the first column on screen,
set zone 9 to 20,
delete column 26
```

```plaintext
SAMPLE FILE A1 V 132 Trunc=132 Size=5 Line=1 Col=8 Alt=2
----->
.<!1....+....>....+....3....+....4....+....5....+....6....+....7....
is an example xample
67890
   12345        345
   67890        678
```

Figure 4: Result of the subcommands above
XEDIT (you can get them by typing HP ? in the XEDIT command line).

Figures 1 to 4 are examples of using the program.

HP XEDIT

/*DOCUSTART */ */
/*PROCEDURE.. : HP */ */
/*LANGUAGE .. : REXX */ */
/*USER ........ : All CMS/XEDIT - Users */ */
/*SYNTAX ...... : HP (from XEDIT commandline) */ */
/*USAGE ...... : Horizontal Prefix line for various commands */ */
/* like Move, Copy, Delete columns */ */
/* Following commands are supported: */ */
/* M,MM,D,DD,C,CC,F,P,",",.,A,I,,<,>,T */ */
/* M ...... Move single column */ */
/* MM ...... Move multiple columns */ */
/* C ...... Copy single column */ */
/* CC ...... Copy multiple columns */ */
/* D ...... Delete single column */ */
/* DD ...... Delete multiple columns */ */
/* " ...... Duplicate single column */ */
/* "" ...... Duplicate multiple columns */ */
/* F ...... target ("Following") column for */ */
/* Move and Copy operation */ */
/* P ...... target ("Preceding") column for */ */
/* Move and Copy operation */ */
/* A ...... Add column (equivalent to I) */ */
/* I ...... Insert column (equivalent to A) */ */
/* < ...... Set left zone */ */
/* > ...... Set right zone */ */
/* T ...... Set Truncation column */ */
/* / ...... Make this column the first on screen */ */
/* The commands for changing columns are valid for */ */
/* all rows from the current line up to EOF. */ */
/* Numeric parameters are not supported at this time. */ */
/*DOCUEND */ */
trace off
arg parm
if parm <> ' '.
then do
 'VMFCLEAR'
do i=1 to 31
   say sourceline(i)
end
exit 1
end
my_scale_nr = '3'                 /* set this number as you need */
my_scale_color = 'YELLOW'         /* set this colour as you like */

'EXTRACT /AUTOSAVE/ZONE/ALT/VERSIFT/EFNAME/RESERVED *
'GLOBALV SELECT HP GET HP'
if hp <> efname.1,
then do                                            /* first call */
  'SET AUTOSAVE OFF'
  'SET CTLCHAR % NOPROTECT' my_scale_color
  'SET CTLCHAR " ESCAPE'
  'GLOBALV SELECT HP SETL HP' efname.1
end
else call stopit
equal = ''
do forever
  call vershift
  'READ ALL NUMBER TAG'
  if rc = 6,
  then call stopit
do queued()
  parse pull tag 5 cmd
  if tag = 'FIL',
  then iterate
  else do
    'EXTRACT /CASE'
    if case.1 = 'UPPER',
    then upper cmd
    parse var cmd line rest
    if tag='CMD'|cmd='=' | (tag='PFK'|tag='PAK')&rest='='. 
    then do
      equal                 /* repeat last command         */
      iterate
    end
    if tag='PFK' | tag='PAK',// PFKey or PAKey
    then do
      equal=rest            /* save command for later use  */
      rest
      iterate
    end
    if tag='CMD',           /* command line                */
    then do
      equal=cmd             /* save command for later use  */
      cmd
      iterate
    end
    if tag='ETK',           /* ENTER Key                   */
    then cmd                 /* execute command             */
    parse upper var rest . text
    if tag='RES' & text<>res,// REServed line modified ! */
then call reserved  /* now here is work to do */
if tag='PRF',       /* normal prefix area */
then do
  parse var text number text
  'EXTRACT /LINE'
  'LOCATE :'number 'LPREFIX' text
  if text <> '/'.
    then 'LOCATE :'line.1
end
end
end
exit

vershift:
  'EXTRACT /VERSHIFT/VERIFY/PREFIX/ZONE/TRUNC/COLUMN/SCALE/LSCREEN'
settings = vershift.1||verify.2||zone.1||zone.2||trunc.1||column.1
settings =
settings||scale.1||scale.2||prefix.1||prefix.2||lscreen.2
if oldsettings <> settings,
  then do
    /* assemble new prefix line */
    res = ''
    from = vershift.1%10
    if vershift.1<0,
      then from = (vershift.1-9)%10
    do i=1 to lscreen.2%10+1
      res = res||'....+'||right(from+i,5,'.'
    end
    res = res||'....+....'
    res = substr(res,((vershift.1+10000)//10)+2,word(verify.2,2)-1)
  call over column.1 '|
  call over zone.1  '<'
  call over zone.2  '>
  call over trunc.1  'T'
  if prefix.1||prefix.2='ONLEFT',/* prefix area on the left */
    then res = '     .'||res
  if scale.1 = 'OFF',
    then scaline = my_scale_nr
  else scaline = scale.2
  'SET RESERVED' scaline 'NOHIGH' '%'||res
  oldsettings = settings
end
return

over:arg where char  /* overlay prefix line */
  where = where-vershift.1-1
  if where < word(verify.2,2) & where > 1,
    then res = overlay(char,res,where,1)
return
reserved:  /* interpret prefix commands */
t = verify(text, 'Ø123456879.+-+-" ACDFIMPT')
if \( t > 0 \),
then return err('Invalid prefix command' substr(text,t,1))
if prefix.1||prefix.2 = 'ONLEFT',
then text = substr(text,6)
else text = '.||text

do until ret <> 1
    call block 'DD' Ø
    /* Delete */
    if result = Ø,
    then call block 'D' Ø
    if result = 1,
    then call pipe '1-from-1' 1' upto+1'* N'
end

do until ret <> 1
    call block 'CC' 1
    /* Copy */
    if result = Ø,
    then call block 'C' 1
    if result = 1,
    then call pipe '1-where' 1' from'-upto 'N' where+1'-* N'
end

do until ret <> 1
    call block 'MM' 1
    /* Move */
    if result = Ø,
    then call block 'M' 1
    if result = 1,
    then do
        if where < from,
        then pip = '1-where '1',
            from'-upto 'N' where+1'-from-1 'N' upto+1'-* N'
        else pip = '1-from-1 '1',
            upto+1'-where 'N' from'-upto 'N' where+1'-* N'
        call pipe pip
    end
end

do until ret <> 1
    call block '""' Ø
    /* Duplicate */
    if result = Ø,
    then call block '"' Ø
    if result = 1,
    then call pipe '1'-upto '1' from'-upto 'N' upto+1||'-* N'
end

do until ret <> 1
    if block('A',Ø)=1,
    /* Insert col */
    then call pipe '1-''from' 1 / / N' from+1'-* N'
end

do until ret <> 1
    if block('I',Ø)=1,
    /* Insert col */
    then call pipe '1-''from' 1 / / N' from+1'-* N'
end

z1 = zone.1
z2 = zone.2

do until ret <> 1
if block('<', Ø) = 1 & zone.1 <> from, /* left zone */
then z1 = from
end

do until ret <> 1
  if block('>', Ø) = 1 & zone.2 <> from, /* right zone */
    then z2 = from
  end
end

'SET ZONE' z1 z2

do until ret <> 1
  if block('T', Ø) = 1 & from <> trunc.1, /* Truncation */
      then SET TRUNC from
  end
end
do until ret <> 1
  if block('/', Ø) = 1, /* Current col*/
      then 'RIGHT' from-vershift.1 - 1
end

return /* ignore invalid prefix cmd */

block: arg action whereto . /* process any prefix command */

from = pos(action, text)
if from > Ø,
  then do
    if (from + vershift.1) < Ø,
      then return err('Invalid position for' action)
    ret = 1
    if length(action) = 2,
      then do
        upto = pos(action, substr(text, from + 2)) + 1
        if upto = 1,
          then return err('Blockend missing for' action)
      end
    else upto = -1
    from = from + vershift.1
    upto = upto + from + 1
    if whereto = 1,
      then do
        f = pos('F', text)
        if f > Ø,
          then do
            where = f
            PIPE(name HP1) VAR TEXT |'.
            XLATE' where 'F . |'.
            'VAR TEXT'
          end
        else do
          p = pos('P', text)
          if p > Ø,
            then do
              where = p - 1
              PIPE(name HP2) VAR TEXT |'.
              XLATE' where 'P . |'.
              'VAR TEXT'
        end
  end
  else do
    ""
end
else do
    where = Ø
    return err('Target missing for' action)
end

where = where + vershift.1
if length(action)=2 & where>from & where<upto,
    then return err('Prefix invalid between block of columns')
end
if from<Ø | upto<Ø | (where<Ø & whereto=1),
    then return err('Input not allowed for negative columns')
'PIPE(name HP3) VAR TEXT |'.
'XLATE' from-vershift.1'-'upto-vershift.1 left(action.1) '. |'.
'VAR TEXT'
end
else ret = Ø
return ret

pipe: arg pip                          /* assemble the lines          */
first = word(pip,1)
if first='1-Ø'|first='1--1'|first='Ø-Ø',
    then pip=delword(pip,1,2)
'EXTRACT /LRECL/LINE/ALT/MSGMODE'
'PIPE(name HP4)   |',            /* apply assembled 'PIPE SPECS */
'XEDIT   |'.            /* subcommand on all lines   */
'PAD' trunc.1 ' |',               /* until EOF                   */
'SPECS' pip ' |',
>'$HP$ CMSUT1 A3'                       /* hope this file doesn't exist */
if rc = Ø, then do
    if vershift.1 < Ø, then do
        spec1 = (1-vershift.1)||'-* 1'
        spec2 = '1-*' 1-vershift.1
    end
    else do
        spec1 = '1-*' 1+vershift.1
        spec2 = (1+vershift.1)||'-* 1'
    end
'PIPE(name HP5)   |',            /* assemble new prefix line    */
'VAR TEXT   |',            /* according the specifications*/
'SPECS' spec1 ' |',
'PAD' trunc.1 ' |',
'SPECS' pip ' |',
'SPECS' spec2 ' |',
'VAR TEXT'
'LOCATE :'line.1
'SET EMSG OFF'
'DELETE *'                     /* this is why HP XEDIT does   */
'GET $HP$ CMSUT1 A3'           /* NOT work properly with      */
'LOCATE :'line.1               /* excluded lines              */
CMS back-up/restore – part 5

This month we conclude the code for the CMS back-up/restore system.

BRCHKLST EXEC

/******************************************************************************
** BRCHKLST - CMS Back-up/Restore create CHeck LiST  
******************************************************************************/
Trace Off
Address COMMAND
Sel. = '
' 'EXECIO * DISKR CMSBR SELECTED A 1 (FINIS'
Do Queued()
    Parse Pull Id Cuu . . . . . Sel.Id.Cuu
End
'EXECIO * DISKR CMSBR SCANLIST A 1 (FINIS MARGINS 1 4Ø'
Do A=1 To Queued() By 1
BRDSIEC EXEC

/**************************************************************/
** BRDSIEC - CMS Back-up/Restore Directory Scan and (if required) **
** Include/Exclude Check                                      **
****************************************************************************/
Address COMMAND
Arg FullBack,Fname,Scan_For

Mdisk. = '?'
'EXEC BRQVDASD'
If Rc = Ø Then Do
  Pull MDisk_List
  Do Queued()
    Pull Volser Mdisk.Volser
  End
End

Address XEDIT 'MACRO BRDCNTRL PF % RUN Reading directory information'
Exclude.Ø = Ø
Include.Ø = Ø
If Fullback = 'NO' Then Do
  'EXECIO * DISKR Fname 'EXCLUDE A 1 (FINIS STRIP'
  'EXECIO * DISKR èALWAYSè EXCLUDE A 1 (FINIS STRIP'
  Exclude.Ø = Queued()
  All_Packs_Excluded = 'N'
  Excl_Ids =
  Excl_Vols=
  Do A=1 To Exclude.Ø By 1
    If Excl_Parm1.A <> 'DASD' Then Excl_Ids = Excl_Ids Excl_Parm1.A
    Else Do
      If Excl_Parm2.A = 'ALL' Then All_Packs_Excluded = 'Y'
      Excl_Vols = Excl_Vols Excl_Parm2.A
    End
  End
  'EXECIO * DISKR Fname 'INCLUDE A 1 (FINIS STRIP'
  'EXECIO * DISKR èALWAYSè INCLUDE A 1 (FINIS STRIP'
  Include.Ø = Queued()
  Incl_Ids =
  Do A=1 To Include.Ø By 1
    If Incl_Parm1.A <> 'DASD' Then Incl_Ids = Incl_Ids Incl_Parm1.A
'BRDIRSCN' Scan_For
If Rc <> Ø Then Do
    Push 'Error reading the directory.'
    Return Rc
End
If Exclude.Ø > Ø Then Do
    Address XEDIT 'MACRO BRDCNTRL PF %',
        'RUN Include/Exclude check running'
End
INCLUDE = 'YES'
Force = '-'
Cyl_Found = Ø
Nr = Ø
Do Queued()
    Parse Pull Userline
    Parse Value Userline With Userid Cuu Volid Cyl_Start . Cyl_Total .
    If Exclude.Ø > Ø Then Do
        Call Excl_Check
        If Incl_Parm3.A = 'FORCE' Then Force = 'FORCE'
        Else Force = '-'
    End
    If Include = 'NO' Then Iterate
    Parse Value Mdisk.Volid With Dasd_Cuu Type Size .
    If Dasd_Cuu = '?' Then Do
        Push 'Volume Volid 'must be EXCLUDED - Not available.'
        Return Sigl
    End
    If Cyl_Total <> Size Then Do
        Cyl_Found = Cyl_Found + Cyl_Total
        Nr = Nr + 1
        Stack.Nr = Right(Dasd_Cuu,4,Ø) Force Userline Type
    End
    Do A=A To Nr By 1
        Queue Stack.A
    End
End
Return Ø Cyl_Found

Excl_Check:
A = Find(Excl_Ids,Userid)
If A <> Ø Then Do
    Exclude = 'USER'
    Do A=A To Exclude.Ø By 1
        If Excl_Parm1.A <> Userid Then Iterate
        If Excl_Parm2.A = Cuu | .
        If Excl_Parm2.A = 'ALL' Then Signal Include_Check
        End
    End
End

© 1997. Xephyon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.
Exclude = 'PACK'
If All_Packs_Excluded = 'Y' Then Signal Include_Check
If Find(Excl_Vols,Valid) > Ø Then Signal Include_Check
Include = 'YES'
Return

Include_Check:
Include = 'NO'
If Include.Ø = Ø Then Return
If Exclude = 'USER' Then Do
  If Find(Incl_Ids,Userid) = Ø Then Return
End
Include = 'YES'
Do A=1 To Include.Ø By 1
  Select
    When Incl_Parm1.A = Userid Then Do
      If Incl_Parm2.A = Cuu |,
        Incl_Parm2.A = 'ALL' Then Return
    End
    When Exclude = 'USER' Then Nop
    When Incl_Parm1.A = 'DASD' Then Do
      If Incl_Parm2.A = Volid Then Return
    End
    Otherwise Nop
  End
Include = 'NO'
Return

BRINST EXEC
/**********************************************************************
** BRINST - CMS Back-up/Restore installation.                      **
**********************************************************************/
Address COMMAND
'CP SET EMSG ON'
"STATE CMSBR COMPTBL A"
If Rc = Ø Then Do
  Say "CMSBR COMPTBL A already created"
  Exit Ø
End
Say "Enter DIRECTORY cuu (As assigned to this user)."
Pull Cuu .
If Rc <> Ø Then Do
  "CP QUERY VIRTUAL" Cuu
  Exit Rc
End
Signal On Error
"EXECIO * DISKR BRDIRSCN ASSEMBLE B 1 (FINIS STEM CUU. FIND /CUU /")
"EXECIO 1 DISKW BRDIRSCN ASSEMBLE B" Word(Cuu.2,2) "(FINIS STRING",

"CUU DC X"Right(Cuu,8,Ø)"") DIRECTRY CUU"

"GLOBAL MACLIB CMSLIB DMSSP DMKSP"
Call Assemble "BRVM20S / NOMOD"
Call Assemble "BRMDSCN"
Call Assemble "BRDIRSCN"
Call Assemble "BRADPSW"
Call Assemble "BRDDRS RLDSAVE"
"ERASE BRVM20S TEXT B1"
"ERASE LOAD MAP A"
Line= "** CMS Back-up/Restore system file - DO NOT ALTER OR DELETE **"
"EXECIO 1 DISKW CMSBR COMPTBL A1 Ø F 2Ø2 (FINIS VAR LINE"
Say
Say "Installation complete"
Exit
Error:
Say
Interpret Say '""Rc="Rc "from"" Sourceline(Sigl)
Exit Rc
Assemble:
Arg Fn LoadOpt "/" GenOpt
"STATE" Fn "ASSEMBLE B1"
"VMFCLEAR"
Say "Assembling" Fn
"ASSEMBLE" Fn "(NOXREF NOLIST NOPRINT"
If GenOpt = "NOMOD" Then Return
"LOAD" Fn "(" LoadOpt
"GENMOD" Fn "MODULE B1 (ALL"
"ERASE" Fn "TEXT B1"
Return

BRLFILE EXEC

/***********************************************************************/
** BRLFILE - CMS Back-up/Restore ListFILE. Used at RESTORE **
************************************************************************/
Address COMMAND
ARG R_CUU R_MODE FN FT SORT_BY
'ERASE CMS EXEC A'
'LISTFILE' Fn Ft R_Mode "(DATE NOHEADER EXEC"
'BRADPSW' Right(R_CUU,8,Ø)
If Rc <> Ø Then Do
Push 'BRADPSW error'
Exit (99ØØØ+Rc)
End

© 1997. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.
'ACCESS' R_Cuu R_Mode
'LISTFILE' Fn Ft R_Mode '(DATE NOHEADER APPEND'
'STATE CMS EXEC A'
If Rc <> Ø Then Exit
Push Sort_By
'XEDIT CMS EXEC A (NOMSG PROFILE BRSORT'
If Rc > 99000 Then Exit Rc
'EXECIO * DISKR CMS EXEC A 1'
If Rc <> Ø Then Do
  Push 'EXECIO error reading sorted listing'
  Exit Rc
End
'ERASE CMS EXEC A'

BRNEWTAP EXEC
/** ************************************************************
** BRNEWTAP - CMS Back-up/Restore. Wait for NEW TApe mount.    **
** ************************************************************/
ADDRESS XEDIT
'EXTRACT /RESERVED */'
Do A=1 To Reserved.Ø By 1
  If Word(Reserved.A,1) <> 21 Then Iterate
  Parse Value Reserved.A With '***' Func Rest '***'
  Leave
End
'MACRO BRD CNTRL RUN Ø- Func 'waiting Ø*** End of tape',
  '- Mount next tape % ALARM'
'MACRO BRD CNTRL ALARM'
Count = Ø
Do Forever
  'CP SLEEP 5 SEC'
  If Word(Diagrc(8,'REW 181'),1) = Ø Then Do
    'MACRO BRD CNTRL RUN' Func Rest
    Exit
  End
  Count = Count + 5
  If Count > 6Ø Then 'MACRO BRD CNTRL ALARM'
End

BRPRINT EXEC
/** *****************************************************
** BRPRINT - CMS Back-up/Restore PRINT. Called twice when printing.**
** *****************************************************
** Parameters: 1st. call: START  2nd. call: END                **
**            **
** BRMAIN prints CONTinuous.          **
**
** The printer is NOT closed by BRMAIN - Do this in the 2nd call.  **

** If this program exits with a non-zero returncode, BRMAIN will **
** CLOSE/PURGE the printer and reply "No list created."

** PRINT and TAG settings are saved and restored by BRMAIN.    **
**********************************************************************/

Arg Func
If Func = 'START' Then  ...your print setup parameters...
If Func = 'END'  Then  ...your print close parameters...

BRQVDASD EXEC

******************************************************************************
** BRQVDASD - CMS Back-up/Restore Query Virtual DASD. **
******************************************************************************
Trace Off
Mdisk. =
Mdisk_List =
Parse Value Diag(8,'QUERY VIRTUAL DASD') With Dasdinfo
Do Until Dasdinfo = ' '  
  Parse Value Dasdinfo With,
      Check Vaddr Type Volser . Size Def_type '15'X Dasdinfo
  If Check <> 'DASD' Then Iterate
  Ok = 'N'
******************************************************************************
** 338Øs can be both 885 and 177Ø cyls big. If the size is 177Ø it  **
** must be a full 338Ø. If it's 885 cyls, assume that it's full.   **
******************************************************************************
Select
  When Type = '338Ø' Then Do
    If Size = 885 & Mdisk.Volser = ' ' Then Ok = 'Y'
    If Size = 177Ø                      Then Ok = 'Y'
  End
  Otherwise Nop
End
If Ok = 'Y' Then Do
  MDisk.Volser = Right(Vaddr,4) Type Size Def_Type
  Mdisk_List = Mdisk_List Volser
End
If Mdisk_List = ' ' Then Exit 1
Queue Mdisk_List
Do A=1 To Words(Mdisk_List) By 1
  X = Word(Mdisk_List,A)
  Queue X Mdisk.X
End
Exit
BRRESALL EXEC

******************************************************************************
** BRRESALL - CMS Back-up/Restore REStore ALL (A full DASD)               **
******************************************************************************

Address COMMAND

'CP SET VMCONIO OFF'
'CP SET CPCONIO OFF'
'CP SET EMSG ON'
'NUCXLOAD BRDDR (SYSTEM SERVICE'
'BRDDR'
'VMFCLEAR'

Say '******************************************************************************'
Say '**                                                             **
Say '** This EXEC restores ALL mini-disks on a specific volume.       **
Say '**                                                             **
Say '** Do not run this EXEC until a full restore of that volume is **
Say '** successfully completed. (Your responsibility).               **
Say '**                                                             **
Say '** Be sure that affected users are logged off the system.        **
Say '******************************************************************************'

Call Reply

Parse Value DiagRc(8,'QUERY VIRTUAL 181') With Rc . . . . Tape_Cuu .
If Rc <> Ø Then Do
    Say 'No tapedrive attached as 181 - Attach and re-run'
    Exit Rc
End

Parse Value BRTAPTYP(181) With Rc Devtype Density
If Rc <> Ø Then Do
    If Rc = 1 Then Say 'Device 181 is not a tapedrive'
    If Rc = 2 Then Say 'Unknown tape drive'
    Exit
End

'TAPE MODESET (DEN' Density

MountFull:
Say 'Mount the "full back-up" tape that was used for the restore.'
Call Reply
Call Tapeload

'EXECIO * DISKR CMSBR SELECTED A 1 (LIFO'
'DESBUF'
If FullBack <> 'YES' Then Do
    'TAPE RUN'
    Say 'Mounted tape was not a "full back-up" tape'
    Say
    Signal MountFull
End
Cyls = C2D(Cyls)
Vol. =
'EXEC BRQVDASD'
Pull Mdisk_List
Do Queued()
  Parse Pull Id Vol.Id
End
If Vol.Volume = ' ' Then Do
  Say 'Volume' Vol.Id 'not available for this user-id'
  Signal Cancel
End
If Word(Diag(8,'QUERY VIRTUAL' Word(Vol.Volume,1)),5) <> 'R/W' Then Do
  Say 'Volume' Vol.Id 'not R/W available. Cuu' Word(Vol.Volume,1)
  Signal Cancel
End
'TAPE RUN'
'ERASE INPUT DDR A'
'DESBUF'
Queue 'INPUT 181' Devtype '(LEAVE MODE' Density
Queue 'OUTPUT' Subword(Vol.Volume,1,2) 'SCRATCH'
Queue 'SYSPRINT CONS'
Queue '
EXECIO 4 DISKW INPUT DDR A Ø F Ø (FINIS'
If Rc <> Ø Then Do
  Say 'Error' Rc 'writing to A-disk'
  Signal Cancel
End
Restored. =
  Say 'About to start restore to' Vol.Id '. Size:' Cyls 'cylinders.'
  Say
MountLim:
  'CONWAIT'
  Say 'Mount a "limited back-up" tape'
  Call Reply
  Call Tapeload
'EXECIO * DISKR CMSBR SELECTED A 1 (STEM SELECT'
  A = Select.Ø
If Word(Select.A,2) <> 'NO' Then Do
  'TAPE RUN'
    Say 'Mounted tape was not a "limited back-up" tape'
    Say
    Signal MountLim
  End
  Select.Ø = Select.Ø - 1
  DDR_Nr = Ø
  TSize  = Ø
  Do A=1 To Select.Ø By 1
    If ChkVol <> Volume Then Iterate
    Restore = 'Y'
    Do B=Scyl To ECyl By 1
If Restored.B = '*' Then Restore = 'N'
End

If Restore = 'Y' Then Do
  Tsize = TSize + Size
  DDR_Nr = DDR_Nr + 1
  DDR.DDR_Nr = 'RESTORE' SCyl 'TO' ECyl
  Do B=Scyl To ECyl By 1
    Restored.B = '*'
  End
End

If DDR_Nr = Ø Then Do
  Say "No mdisks to be restored from this tape"
  Signal Unload
End

Total = Ø
Do A=1 To DDR_Nr By 1
  Parse Value DDR.A With . SCyl . ECyl .
  Size = (ECyl - Scyl) + 1
  Say 'Restoring mdisk' A 'of' DDR.Nr'.',
  'Cylinders restored:' Total 'of' TSize
  'EXECIO 1 DISKW INPUT DDR A 4 (FINIS STRING' DDR.A
  If Rc <> Ø Then Do
    Say 'Error' Rc 'writing to A disk'
    Signal Cancel
  End
  Say
  Say 'Positioning tape ...'
  'CONWAIT'
  'BRDDR BRRESALL' Volume Right(Word(DDR.A,2),4,Ø)
  If Rc <> Ø Then Do
    Say 'Error' Rc 'from BRDDR'
    Signal Cancel
  End
  'VMFCLEAR'
  Total = Total + Size
End

Unload:
  'TAPE RUN'
  Do A=1 To Cyls By 1
    If Restored.A = ' ' Then Leave
  End
  If A > Cyls Then Do
    Say 'All cylinders now successfully restored (updated).'
    Signal Cancel
  End

Say 'When all "limited back-up" tapes have been processed reply CANCEL'
Say
Signal MountLim
Reply:
   'DESBUF'
   'CONWAIT'
   Say
   Do Forever
     Say 'Enter GO to continue or CANCEL to terminate.'
     Parse Upper External Answer
     If Answer = 'CANCEL' Then Signal Cancel
     If Answer = 'GO' Then Leave
   End
   'VMFCLEAR'
   Return

Tapeload:
   Say 'Loading tape information ...'
   Parse Value BRTAPCHK('LOAD') With Rx .
   'VMFCLEAR'
   If Rx = Ø Then Return
   'TAPE RUN'
   Say 'Wrong tape mounted - Load correct tape.'
   Call Reply
   Signal Tapeload

Cancel:
   Say
   Say 'EXEC terminated'
   Say
   Say 'NOTE: If you re-run the EXEC you must mount ALL tapes again.'
   'NUCXDROP BRDDR'
   Exit

BRTAPCHK EXEC

/**/
Address XEDIT 'BRDCNTRL RUN Verifying tape'
Call Clear
Vol1 = 'VOL1'
'TAPE DVOL1 (REWIND'
Rx = Rc
If Rx = Ø Then Do
  'FILEDEF IN TAPI'
  'FILEDEF OUT DISK TAPE HEADER A'
  'MOVEFILE IN OUT'
  'TAPE REW'
  'EXECIO 2 DISKR TAPE HEADER A 1 (FINIS STEM HEADER.'
  'ERASE TAPE HEADER A'
  Vol1 = Substr(Header.1,5,6)
  Hdr1 = Substr(Header.2,5,8)
  If Substr(Header.2,1,4) <> 'HDR1' Then Do
    Rx = 99
    Signal Exit
  End
End
If Function = 'VERIFY' Then Do
  If Hdr1 <> 'CMSBROUT' Then Rx = 1
  Signal Exit
End
If Function = 'DUMP' Then Do
  If Rx <> Ø Then Do
    Rx = 99
    Signal Exit
  End
  'FILEDEF IN DISK CMSBR SELECTED A'
  'FILEDEF OUT TAPI SL 1 VOLID' Vol1 '(NOEOV'
  'LABELDEF OUT FID CMSBROUT'
  'MOVEFILE IN OUT'
  RX = RC
  Signal Exit
End
If Function = 'LOAD' Then Do
  If Rx <> Ø Then Signal Exit
  If Hdr1 <> 'CMSBROUT' Then Do
    Rx = 1
    Signal Exit
  End
  Address XEDIT 'BRDCNTRL RUN Loading information'
  'FILEDEF IN TAPI SL 1'
  'FILEDEF OUT DISK CMSBR SELECTED A'
  'MOVEFILE IN OUT'
  RX = RC
  Signal Exit
End
Exit:
  Call Clear
If Cmstype = 1 Then 'SET CMSTYPE RT'
Return Rx Vol1 Hdr1
Clear:
'LABELDEF * CLEAR'
'FILEDEF * CLEAR'
Return

BRTAPTYP EXEC

**************************************************************************
** BRTAPTYP - CMS Back-up/Restore TApe TYPe.                        **
** ---------------------------------------------------------------- **
** Input: Virtual address of a tape drive.                         **
** Output: Rc DDR_Devicetype Density                                **
** ****
** NOTE: Devtype = 'Ø8'X is a 341Ø but must be specified as a 342Ø. **
**************************************************************************
Parse Value Diag(24,Arg(1)) With . 1 Rdevtypc 2 Rdevtype 3 .
If Rdevtypc <> 'Ø8'X Then Return 1
If Rdevtype  = 'Ø1'X Then Return 'Ø 348Ø 38K'
If Rdevtype  = 'Ø2'X Then Return 'Ø 343Ø 16ØØ'
If Rdevtype  = 'Ø4'X Then Return 'Ø 343Ø 16ØØ'
If Rdevtype  = 'Ø5'X Then Return 'Ø 348Ø 625Ø'
If Rdevtype  = 'Ø6'X Then Return 'Ø 242Ø 16ØØ'
If Rdevtype  = 'Ø7'X Then Return 'Ø 2415 16ØØ'
If Rdevtype  = 'Ø8'X Then Return 'Ø 24Ø1 16ØØ'
Return 2

BRTDISK EXEC

**************************************************************************
** BRTDISK - CMS Back-up/Restore Temporary DISK                     **
**************************************************************************
Address COMMAND
Arg Request Parms
Cuu  = '335'
Mode = 'G'
If Request = 'DEFINE' Then Do
Parse Value Parms With Type Size .
Parse Value DiagRc(8,'DEFINE T'Type Cuu Size) With Rc . Text '15'X
Return Strip(Rc) Strip(Text)
End
If Request = 'ACCESS' Then Do
'ACCESS' Cuu Mode
If Rc <> Ø Then Return Rc
'DESBUF'
'QUERY DISK' Mode '(STACK'

© 1997. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.
Pull.
Pull........Files.
'DESBUF'
Return Ø Files
End
If Request = 'DETACH' Then Do
'RELEASE' Mode
Return DiagRc(8,'DETACH' Cuu)
End
If Request = 'QCUU' Then Return Cuu Mode

CMSBR EXEC

/**************************************************************************
** CMSBR - Start CMS Back-up/Restore program                          **
**************************************************************************/
Trace Off
Arg Option
'DESBUF'
'STATE CMSBR COMPTBL A'
If Rc <> Ø Then Exit Rc
'ACCESS 199 B'
If Rc <> Ø Then Exit Rc
Parse Value Diag(8,'QUERY SET') With,
   'MSG' Msg '.',' 'EMSG' Emsg '.',' 'IMSG' Imsg '.','
'CP SET MSG     OFF'
'CP SET IMSG    OFF'
'CP SET EMSG    IUCV'
'CP SET CPCONIO IUCV'
'CP SET VMCONIO IUCV'
'NUCXLOAD BRDDR (SYSTEM SERVICE'
'BRDDR'
'EXECLOAD BRDCNTRL XEDIT'
Push Option
'XEDIT CMSBR' Userid() 'A (NOMSG PROFILE BRMAIN'
Rx = Rc
'CP SET MSG' Msg
'CP SET IMSG' Imsg
'CP SET EMSG' Emsg
'CP SET CPCONIO OFF'
'CP SET VMCONIO OFF'
'NUCXDROP BRDDR'
'EXECDROP * * (USER'
Do Queued()
   Parse Pull Msg
   Say Msg
End
Exit Rx

Michael Plannthin (Denmark) © Xephon 1997
Dynamic menus system for CMS

The first three installments of *Dynamic menus system for CMS* were published in the January, February, and March issues of *VM Update*. This month we start publication of the on-line administration part.

The on-line administration utilities are used to:

- Administer passwords
- Create menus
- Update menus
- Delete menus
- Update authorizations
- Create/update MENPROCs
- Delete MENPROCs.

To enter the menu system on-line administration, you have to execute OAMENU.

**HMENU.DAT**

THE MENU SYSTEM ON-LINE ADMINISTRATION HAS 6 FUNCTIONS:

_ THE AUTHORIZATIONS MANAGEMENT FUNCTION
_ THE MENUS UPDATE FUNCTION
_ THE MENUS CREATION FUNCTION
_ THE PROCEDURES MANAGEMENT FUNCTION
_ THE ACCESS PASSWORDS DISPLAY FUNCTION
_ THE HELP FACILITY.

1_______________________________________________________________MORE...

THE AUTHORIZATIONS MANAGEMENT FUNCTION

THIS FUNCTION ALLOWS THE MENU SYSTEM ADMINISTRATOR TO SET UP THE APPLICATIONS EACH USER WILL BE ALLOWED TO USE FROM HIS MENUS. THE AUTHORIZATIONS ARE SET BY MENU FOR EACH USER. CALLING THIS FUNCTION BRINGS UP A FULLSCREEN PANEL, WHERE YOU MUST TYPE IN THE NAME OF THE USER YOU WANT TO AUTHORIZE AND THE NAME OF THE MENU TO BE AUTHORIZED.
YOU MAY SCROLL THROUGH THE EXISTING USERS LIST WITH PFØ2 (FORWARD) OR PFØ3 (BACKWARD), AND THROUGH THE EXISTING MENUS LIST WITH PF1Ø OR PF11. TO ALLOW USE OF A MENU LINE FOR THE USER, YOU MUST FILL IN AN 'X' IN THE INPUT FIELD PRECEDING THE MENU LINE DESCRIPTION. IF YOU WANT TO DISALLOW THE USE OF A MENU LINE FOR THE USER, THE INPUT FIELD PRECEDING THE MENU LINE DESCRIPTION MUST BE BLANK.

IF YOU WANT THE AUTHORIZATION MATRIX TO BE UPDATED WITH THE DATA YOU JUST TYPED IN, YOU MUST CHANGE THE UPDATE FLAG TO 'Y' AT THE BOTTOM OF THE PANEL.
TO RESET THE INPUT AND CLEAR THE PANEL, PRESS "CLEAR" OR "PA2". THE EXIT KEY IS SET TO PF12/PF24.

THE MENUS UPDATE FUNCTION.
YOU USE THIS FUNCTION TO UPDATE AN EXISTING MENU. YOU MUST SUPPLY THE NAME OF THE MENU YOU NEED TO UPDATE OR PRESS PF1Ø/PF11 TO SCROLL THROUGH THE LIST OF EXISTING MENUS.
YOU UPDATE A MENU BY TYPING A COMMAND IN THE COMMAND FIELD. VALID COMMANDS ARE: 'A' TO ADD A NEW MENU LINE, 'D' TO DELETE A MENU LINE, AND 'C' TO CHANGE MENU LINE CONTENTS.

TO UPDATE A MENU WITH 'A' OR 'C' COMMAND, YOU MUST TYPE THE NEW DATA OVER THE EXISTING MENU LINE. TO DELETE A LINE, JUST TYPE 'D' IN THE COMMAND FIELD. IN BOTH CASES, THE UPDATE FLAG AT THE BOTTOM OF THE PANEL MUST BE SET TO "Y".
TO RESET THE INPUT AND CLEAR THE PANEL, PRESS "CLEAR" OR "PA2". THE EXIT KEY IS SET TO PF12/PF24.
USE PFØ7/PFØ8 TO SCROLL BACK/FORWARD THROUGH THE MENU LINES LIST.

TO DELETE THE WHOLE MENU, SET THE UPDATE FLAG TO "Y" AND PRESS PFØ6. IF YOU TRY TO UPDATE A NON-EXISTENT MENU, YOU WILL BE ASKED TO CREATE IT FIRST.
TO CREATE A NEW MENU, TYPE IN THE NEW MENU NAME IN THE PROPER FIELD AND PRESS PFØ9.
THE MENUS CREATION FUNCTION

YOU ENTER THIS FUNCTION BY PRESSING PFØ9 FROM THE MENUS UPDATE PANEL. YOU MUST SUPPLY THE MENU DESCRIPTION, FILL IN AT LEAST ONE MENU LINE DEFINITION (DESCRIPTION, EXEC NAME, AND EXEC TYPE), AND, OF COURSE, SET THE UPDATE FLAG TO "Y" AT THE BOTTOM OF THE PANEL. TO RESET THE INPUT AND CLEAR THE PANEL, PRESS "CLEAR" OR "PA2". THE EXIT KEY IS SET TO PF12/ PF24.

THE PROCEDURE MANAGEMENT FUNCTION

THIS FUNCTION IS CALLED TO ADD, UPDATE, AND DELETE PROCEDURES. YOU MUST SUPPLY THE PROCEDURE NAME OR SCROLL THROUGH THE EXISTING PROCEDURES LIST WITH PF1Ø/PF11. TO CHANGE A PROCEDURE LINE, JUST TYPE OVER THE NEW DATA. TO ADD A NEW PROCEDURE LINE FILL IN THE DATA IN AN EMPTY LINE IN THE PANEL. TO DELETE A PROCEDURE LINE, JUST ERASE ALL THE LINE CONTAINS.

TO DELETE THE WHOLE PROCEDURE, PRESS PFØ6. IN ANY CASE, NO CHANGE WILL BE PERFORMED UNLESS THE UPDATE FLAG IS SET TO "Y". TO RESET THE INPUT AND CLEAR THE PANEL, PRESS "CLEAR" OR "PA2". THE EXIT KEY IS SET TO PF12/ PF24.

THE ACCESS PASSWORDS DISPLAY FUNCTION

THIS FUNCTION ALLOWS YOU TO BROWSE THROUGH THE USERS/PASSWORDS LIST IN THE SECURITY PART OF THE DYNAMIC MENUS SYSTEM. CHANGES ARE NOT ALLOWED (FOR NOW!). USE PFØ7/PFØ8 TO SCROLL THROUGH THE USERS LIST. THE EXIT KEY IS SET TO PF12/ PF24.
HMENU EXEC

/* HMENU EXEC */
/* AIDKEY EQUATES */
$F1='PFØ1';$F2='PFØ2';$F3='PFØ3';$F4='PFØ4';$F5='PFØ5';$F6='PFØ6'
$F7='PFØ7';$F8='PFØ8';$F9='PFØ9';$7A='PF1Ø';$7B='PF11';$7C='PF12'
$7D='PF13';$7E='PF14';$7F='PF15';$C4='PF16';$C5='PF17';$C6='PF18'
$C7='PF19';$C8='PF2Ø';$C9='PF21';$4A='PF22';$4B='PF23';$4C='PF24'
$Ø1='PA1';$6E='PA2';$7D='ENTER';$6D='CLEAR'

'SET LANGUAGE (ADD TAF USER' /* MESSAGE REPOSITORY IS TAFUME REPOS */
'VMFCLEAR'
'PIPE CMS L HMENU DATA A (L | DROP 1 | SPECS W6 STRIP 1 | VAR LINES'

RESET_ALL:
MESSAGE.1=COPIES(' ',56)
LINE.=''
DROP=Ø
LOAD_ARRAYS:
'PIPE < HMENU DATA '
'| DROP 'DROP,
'| TAKE 16 '.
'| STEM LINE.'
HMENU :
SCREEN='8ØØ3'X||,
'11Ø0Ø29Ø3CØ2041Ø042F3'X||,'11Ø041E1DØF0'X||,'11Ø0511DFØ'X||,
'11Ø06329Ø3CØ2041F242F5'X||,' DYNAMIC MENUS ADMINISTRATION UTILITIES
'| '11Ø08C1D1DFØ'X||,
'11Ø09D29Ø3CØ2041Ø042F3'X||,'11Ø09F1DFØ'X||,'11Ø0E1DFØ'X||,'11Ø0F1DFØ'X||,
'11Ø10E29Ø3CØ2041F242F5'X||,' HELP FACILITY '||'11Ø11E1DFØ'X||,
'11Ø13D29Ø3CØ2041Ø042F3'X||,'11Ø13F1DFØ'X||,'11Ø18F1DFØ'X||,
'11Ø18F29Ø3CØ2041Ø042F3'X||,'11Ø1911DFØ'X||,
'11Ø19329Ø3CØ2041F242F7'X||'LINE.1 ||'11Ø1D11DFØ'X||,'11Ø1D29Ø3CØ2041Ø042F3'X||,'11Ø1D29Ø3CØ2041Ø042F3'X||,'11Ø1D29Ø3CØ2041Ø042F3'X||,'11Ø1E21DFØ'X||,'11Ø1E29Ø3CØ2041F242F7'X||'LINE.2 ||'11Ø22B1DFØ'X||,'11Ø22D29Ø3CØ2041Ø042F3'X||,'11Ø22F29Ø3CØ2041Ø042F3'X||,'11Ø2311DFØ'X||,'11Ø2329Ø3CØ2041F242F7'X||'LINE.3 ||'11Ø27B1DFØ'X||,'11Ø27D29Ø3CØ2041Ø042F3'X||,'11Ø27F29Ø3CØ2041Ø042F3'X||,'11Ø2811DFØ'X||,'11Ø28329Ø3CØ2041F242F7'X||'LINE.4 ||'11Ø2CB1DFØ'X||,'11Ø2CD29Ø3CØ2041Ø042F3'X||,'11Ø2CF29Ø3CØ2041Ø042F3'X||,'11Ø2D11DFØ'X||,'11Ø2D329Ø3CØ2041F242F7'X||'LINE.5 ||'11Ø31B1DFØ'X||,'11Ø31D29Ø3CØ2041Ø042F3'X||,'11Ø31F29Ø3CØ2041Ø042F3'X||,'11Ø3211DFØ'X||,'11Ø32329Ø3CØ2041F242F7'X||'LINE.6 ||'11Ø36B1DFØ'X||,
MESSAGE.1=COPIES(' ',50)
SELECT
WHEN VALUE(AIDKEY)='PF02' THEN DO /* TOP */
  DROP=0
  LINE='' 'XMITMSG 10 (APPLID TAF CALLER HME NOCOMP VAR'
  SIGNAL_LOAD_ARRAYS
WHEN VALUE(AIDKEY)='PF03' THEN DO  /* BOTTOM */
   DROP=LINES-16
   LINE=''
   'XMITMSG 11 (APPLID TAF CALLER HME NOCOMP VAR'
   SIGNAL LOAD_ARRAYS
   END  /* END PF03 */
WHEN VALUE(AIDKEY)='PF07' THEN DO  /* SCROLL BACKWARD */
   DROP=DROP-16
   IF DROP<=Ø THEN DO
      DROP=Ø
   'XMITMSG 10 (APPLID TAF CALLER HME NOCOMP VAR'
   END
   LINE=''
   SIGNAL LOAD_ARRAYS
   END  /* END PF07 */
WHEN VALUE(AIDKEY)='PF08' THEN DO  /* SCROLL FORWARD */
   DROP=DROP+16
   IF DROP>=LINES THEN DO
      DROP=LINES-16
      IF DROP<Ø THEN DROP=Ø
   'XMITMSG 11 (APPLID TAF CALLER HME NOCOMP VAR'
   END
   LINE=''
   SIGNAL LOAD_ARRAYS
   END  /* END PF08 */
WHEN VALUE(AIDKEY)='PF13' THEN DO  /* TEST */
   SAY MENU
   SIGNAL HMENU
   END  /* END PF13 TEST */
WHEN VALUE(AIDKEY)='PF24' | VALUE(AIDKEY)='PF12' THEN EXIT
WHEN VALUE(AIDKEY)='CLEAR' | VALUE(AIDKEY)='PA2' THEN SIGNAL RESET_ALL
WHEN VALUE(AIDKEY)='ENTER' THEN SIGNAL HMENU
   OTHERWISE DO
      'XMITMSG 1 'VALUE(AIDKEY)' (APPLID TAF CALLER HME NOCOMP VAR'
      SIGNAL HMENU
      END  /* END OTHERWISE */
RETURN
/*
*/

OAMENU EXEC

   /* OAMENU EXEC */
   'VMFCLEAR'
   TITLE=CENTER('DYNAMIC MENUS SYSTEM',26)
   MENU_TITLE=CENTER('YOUR COMPANY NAME',26)
   'SET LANGUAGE (ADD TAF USER '

OAMENU_START:
'\"SET CMSTYPE HT\";'\"ACC \"DATA_FILES_DISK\" X/A\";'\"SET CMSTYPE RT\"'
$F1='PF01'; $F2='PF02'; $F3='PF03'; $F4='PF04'; $F5='PF05'; $F6='PF06'
$F7='PF07'; $F8='PF08'; $F9='PF09'; $7A='PF10'; $7B='PF11'; $7C='PF12'
$7D='PF13'; $7E='PF14'; $7F='PF15'; $8A='PF16'; $8B='PF17'; $8C='PF18'
$8D='PF19'; $8E='PF20'; $9A='PF21'; $9B='PF22'; $9C='PF23'; $9D='PF24'
$7E='PA1'; $7F='PA2'; $7D='CLEAR'
ERROR_MESSAGE=COPIES(' ',63)
B2_LINE.1='1.HELP'
B2_LINE.2='2.AUTHORIZATIONS'
B2_LINE.3='3.MENUS'
B2_LINE.4='4.PROCEDURES'
B2_LINE.5='5.PASSWORDS DISPLAY'
B2_LINE.6='6.EXIT'
BLOCK2 =
'11024B2903C02041F242F6'X'||'1102641DF0'X'||
'11029B2903C02041F242F6'X'||'1102A2903C030410042F6'X'||'ONLINE ADMN MENU '||
'2841F2'X'||'284100'X'||'1102B41DF0'X'||
'1102EB2903C02041F242F6'X'||'1103041DF0'X'||
'11033B2903C02041F242F6'X'||'1103541DF0'X'||
'11038B2903C02041F242F6'X'||'1103A41DF0'X'||
'1103DB2903C02041F242F6'X'||'1103F41DF0'X'||
'11042B2903C02041F242F6'X'||'1104441DF0'X'||
'11047B2903C02041F242F6'X'||'1104941DF0'X'||
'1104CB2903C02041F242F6'X'||'1104E41DF0'X'||
'11051B2903C02041F242F6'X'||'1105341DF0'X'||
'11056B2903C02041F242F6'X'||'1105741DF0'X'||
'1105BB2903C02041F242F6'X'||'1105D41DF0'X'||
'11060B2903C02041F242F6'X'||'1106241DF0'X
OAMENU :
SCREEN='B003'X||,
'110002903C020410042F6'X||,
'\"\"
\textbackslash{}11010A2903C02041F242F50'X||TITLE'||'1101251DF0'X'||.
\textbackslash{}11013E2903C020410042F60'X'||'1101401DF0'X'||.
\textbackslash{}1101402903C020410042F60'X||.

\textbackslash{}1101901DF0'X||.
\textbackslash{}1101902903C020410042F60'X||'1101921DF0'X||.
\textbackslash{}1101DE2903C020410042F60'X||'1101E01DF0'X||.
\textbackslash{}1101E02903C020410042F60'X||'1101E21DF0'X||.
\textbackslash{}11022E2903C020410042F60'X||'1102301DF0'X||.
\textbackslash{}1102302903C020410042F60'X||'1102321DF0'X||.
\textbackslash{}11027E2903C020410042F60'X||'1102801DF0'X||.
\textbackslash{}1102802903C020410042F60'X||'1102821DF0'X||.
\textbackslash{}1102E2903C020410042F60'X||'1102D01DF0'X||.
\textbackslash{}1102D02903C020410042F60'X||'1102D21DF0'X||.
\textbackslash{}11031E2903C020410042F60'X||'1103201DF0'X||.
\textbackslash{}1103202903C020410042F60'X||'1103221DF0'X||.
\textbackslash{}11036E2903C020410042F60'X||'1103701DF0'X||.
\textbackslash{}1103702903C020410042F60'X||'1103721DF0'X||.
\textbackslash{}1103B2903C020410042F60'X||'1103C01DF0'X||.
\textbackslash{}1103C02903C020410042F60'X||'1103C12DF0'X||.
\textbackslash{}11046E2903C020410042F60'X||'1104101DF0'X||.
\textbackslash{}1104102903C020410042F60'X||'1104121DF0'X||.
\textbackslash{}11045E2903C020410042F60'X||'1104601DF0'X||.
\textbackslash{}1104602903C020410042F60'X||'1104621DF0'X||.
\textbackslash{}1104AE2903C020410042F60'X||'1104B01DF0'X||.
\textbackslash{}1104B02903C020410042F60'X||'1104B21DF0'X||.
\textbackslash{}1104F2903C020410042F60'X||'1105001DF0'X||.
\textbackslash{}110502903C020410042F60'X||'1105021DF0'X||.
\textbackslash{}11054E2903C020410042F60'X||'1105501DF0'X||.
\textbackslash{}1105502903C020410042F60'X||'1105521DF0'X||.
\textbackslash{}11059E2903C020410042F60'X||'1105A01DF0'X||.
\textbackslash{}1105A02903C020410042F60'X||'1105A21DF0'X||.
\textbackslash{}1105E2903C020410042F60'X||'1105F01DF0'X||.
\textbackslash{}1105F02903C020410042F60'X||'1105F21DF0'X||.
\textbackslash{}11063E2903C020410042F60'X||'1106401DF0'X||.
\textbackslash{}1106402903C020410042F60'X||'1106421DF0'X||.
\textbackslash{}11068E2903C020410042F60'X||'1106901DF0'X||.
\textbackslash{}1106902903C020410042F60'X||'1106921DF0'X||.
\textbackslash{}1106D2903C020410042F60'X||'1106E01DF0'X||.
\textbackslash{}1106E2903C020410042F60'X||'1106E21DF0'X||.
\textbackslash{}1106ED2903C02041F42F20'X||ERROR_MESSAGE||'11072D1DF0'X||.
\textbackslash{}11072E2903C020410042F60'X||'1107301DF0'X||.
\textbackslash{}1107302903C020410042F60'X||.'

'\textbackslash{}'\textbackslash{}11019F130'X
PIPE (ENCHAR ?) VAR SCREEN | FULLSCREEN '
'\textbackslash{}' SPLIT AT ANYOF /''11''X/'/',
'\textbackslash{}' A: FANOUT',

SELECT /* CHECK AID KEY PRESSED */
  WHEN VALUE(AIDKEY)='PF06',
    | VALUE(AIDKEY)='PF12',
    | VALUE(AIDKEY)='PF24' THEN EXIT /* EXIT WITH PF06/12/24 */
  WHEN VALUE(AIDKEY)='CLEAR' THEN SIGNAL OAMENU_START
  WHEN VALUE(AIDKEY)='PF01' THEN DO /* HELP */
    CALL HMENU
    SIGNAL OAMENU
  END /* END PF01 */
  WHEN VALUE(AIDKEY)='PF02' THEN DO /* AUTHORIZATIONS */
    CALL XAUTH
    SIGNAL OAMENU
  END /* END PF02 */
  WHEN VALUE(AIDKEY)='PF03' THEN DO /* MENUS */
    CALL XLINES
    SIGNAL OAMENU
  END /* END PF03 */
  WHEN VALUE(AIDKEY)='PF04' THEN DO /* PROCEDURES */
    CALL XSTATMS
    SIGNAL OAMENU
  END /* END PF04 */
  WHEN VALUE(AIDKEY)='PF05' THEN DO /* PASSWORDS */
    CALL XPASW
    SIGNAL OAMENU
  END /* END PF05 */
  WHEN VALUE(AIDKEY)='ENTER' THEN DO /* PROCESS ENTER */
    IF COM™='' THEN DO
      IF COM>6 THEN DO
        BLOCK_TITLE='ONLINE ADMIN MENU'
        'XMITMSG 5 BLOCK_TITLE COM (APPLID TAF CALLER OAM NOCOMP VAR'
        ERROR_MESSAGE=MESSAGE.1
        SIGNAL OAMENU
        END
      ELSE DO /* PROCESS OPTION */
      END
    SELECT
      WHEN COM=1 THEN CALL HMENU
      WHEN COM=2 THEN CALL XAUTH
      WHEN COM=3 THEN CALL XLINES
      WHEN COM=4 THEN CALL XSTATMS
      WHEN COM=5 THEN CALL XPASW
      WHEN COM=6 THEN CALL GOOUT
      OTHERWISE NOP
    END
\[\text{Call for papers}\]

Why not share your expertise and earn money at the same time? *VM Update* is looking for REXX EXECs, macros, program code, etc, that experienced VMers have written to make their life, or the lives of their users, easier. We will publish it (after vetting by our expert panel) and send you a cheque when the article is published. Articles can be of any length and can be sent or e-mailed to Trevor Eddolls at any of the addresses shown on page 2. Why not call now for a free copy of our *Notes for contributors*?
New from Xephon is *New directions in VM/VSE*. This is the latest in a series of reports and contains chapters on: the comparison of CMS programming interfaces—the program stack, CMS Pipelines, and CSL; VM Web servers; the use of intranets; migrating from OfficeVision; VM/VSE to MVS migration; re-engineering COBOL applications; and future directions for VM. It costs £175.00 ($265.00).

For further information contact:
Xephon, 27-35 London Road, Newbury, Berks, RG14 1JL, UK.
Tel: (01635) 33823.
Xephon, 1301 West Highway 407, Suite 201-450, Lewisville, TX 75067, USA.
Tel: (940) 455 7050.

**IBM** is celebrating VM’s 25th birthday this year. There is more information about events in your area from IBM’s VM Web page – [http://www.vm.ibm.com](http://www.vm.ibm.com).

For further information contact your local IBM representative.

***